NASA Aeronautics Research Institute

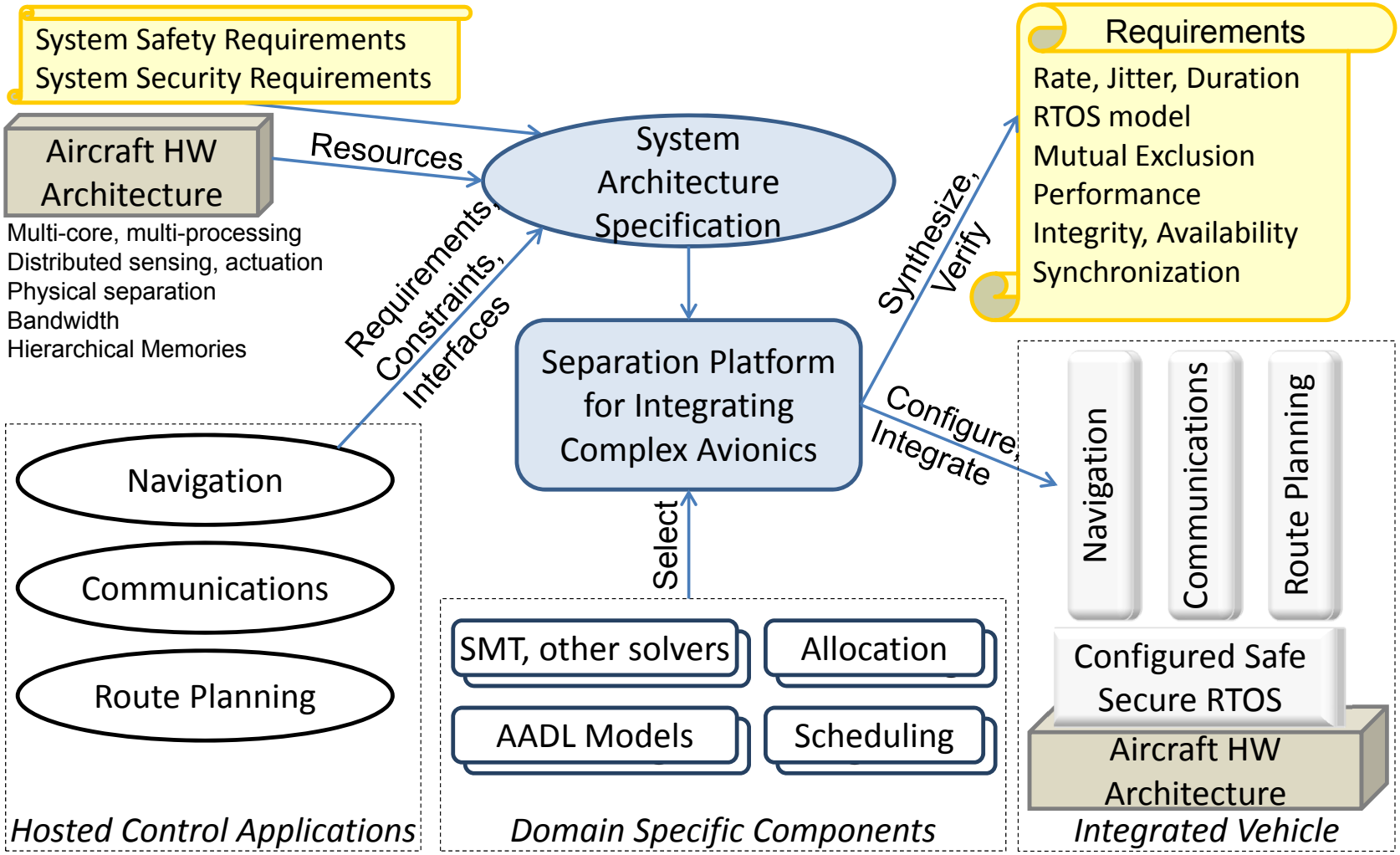
Separation Platform for Integrating Complex Avionics (SPICA)

NASA Aeronautics Research Mission Directorate (ARMD)

FY12 LEARN Phase I Technical Seminar

November 13–15, 2013

SPICA


System Safety Requirements
System Security Requirements
Aircraft HW Architecture
Multi-core, multi-processing
Distributed sensing, actuation
Physical separation
Bandwidth
Hierarchical Memories
Resources
System Architecture Specification
Requirements, Constraints, Interfaces
Synthesize, Verify
Requirements
Rate, Jitter, Duration
RTOS model
Mutual Exclusion
Performance
Integrity, Availability
Synchronization
Separation Platform for Integrating Complex Avionics
Configure, Integrate
Navigation
Communications
Route Planning
Select
SMT, other solvers
Allocation
AADL Models
Scheduling
Navigation
Communications
Route Planning
Configured Safe Secure RTOS
Aircraft HW Architecture
Hosted Control Applications
Domain Specific Components
Integrated Vehicle

Presentation Outline

- The problem
- Technical approach
- Results of Phase I
- Impact
- Next steps

787 Common Data Network

Diagram showing where the common core system (CCS) is connected throughout the 787 aircraft. Most of what is noted in the fuselage are the 21 or so remote data concentrators that GE provides and are advanced sensors to the CCS. Source: GE Aviation.

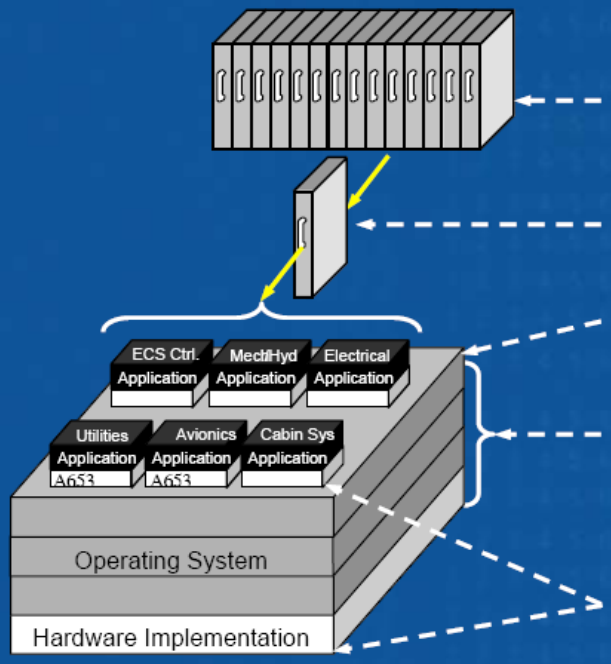
Common Core System Benefits

Common Data Network

- Open industry standard interfaces A664
- Eliminate multiple standards & protocols
- Fiber Optic Network media

Common Computing Resource

- Based on Open System Architecture Principles



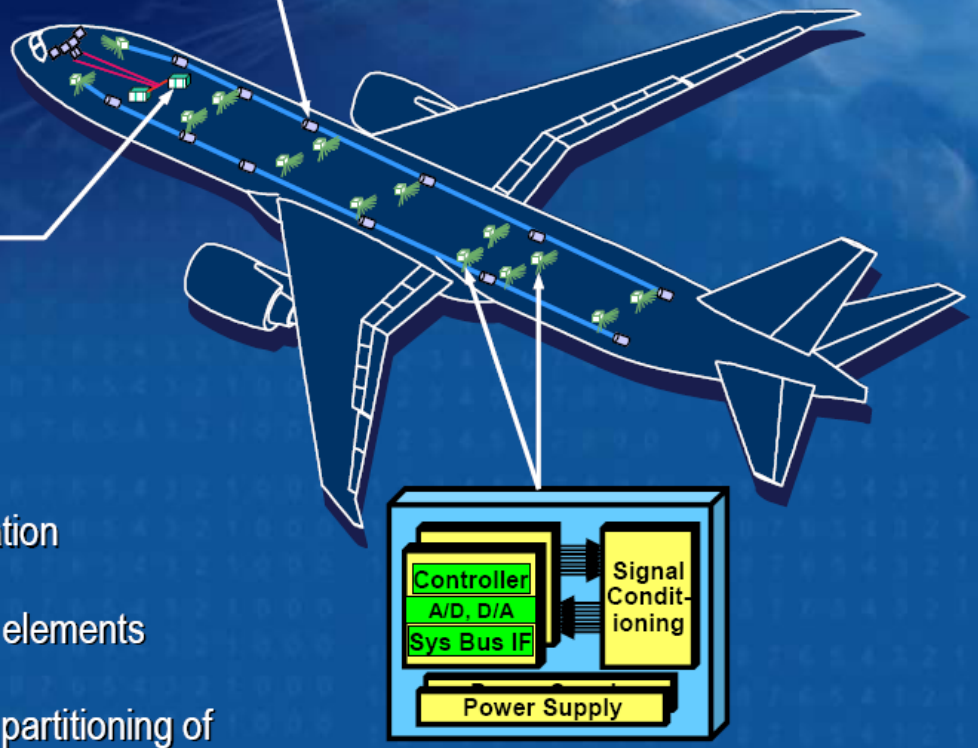
Modular Implementation

of common elements

with robust partitioning of functions in software

hierarchical layering of services

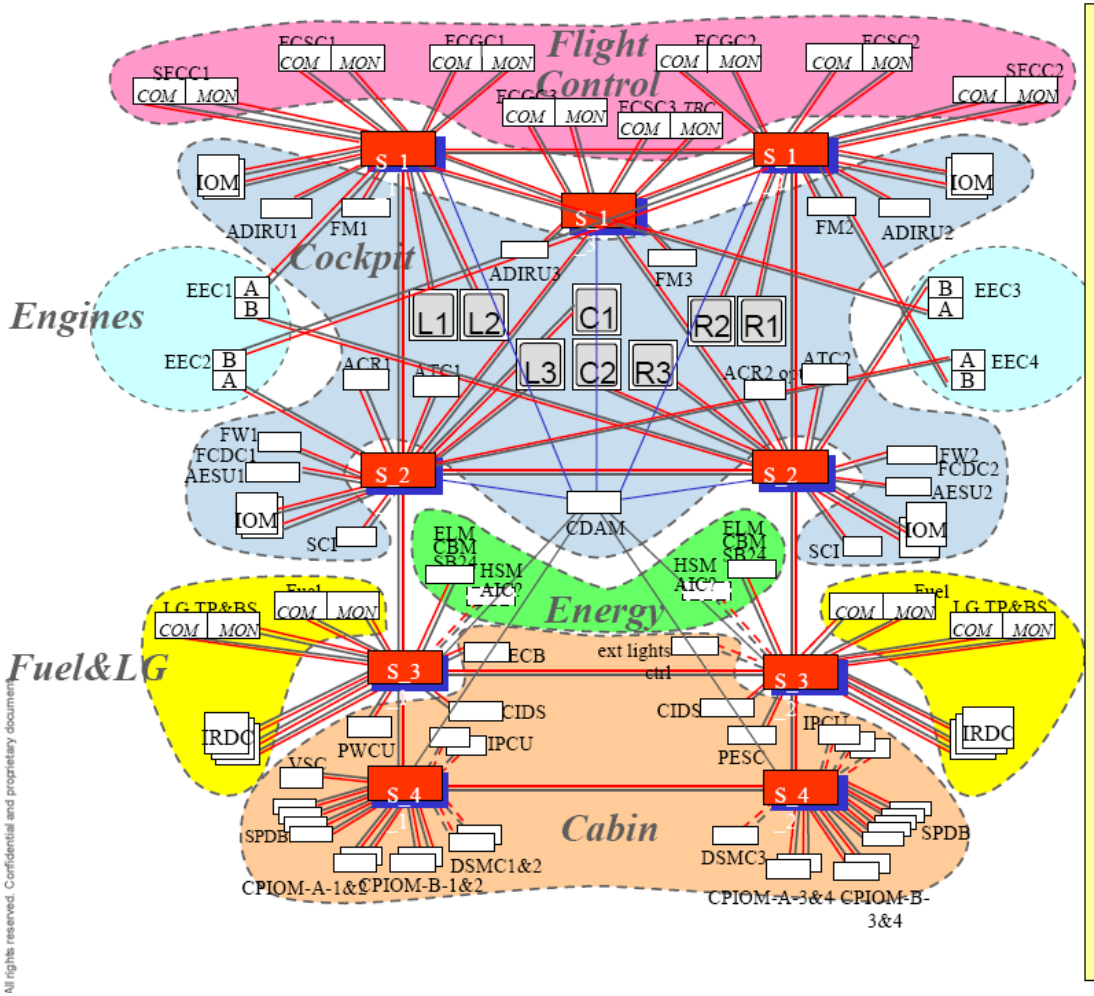
having well defined, standardized, rigidly enforced key interfaces A653



Remote Data Concentrators

- Reduces airplane wiring/weight,
- Ease of system upgrade/modification
- Highly reliable

A380 IMA

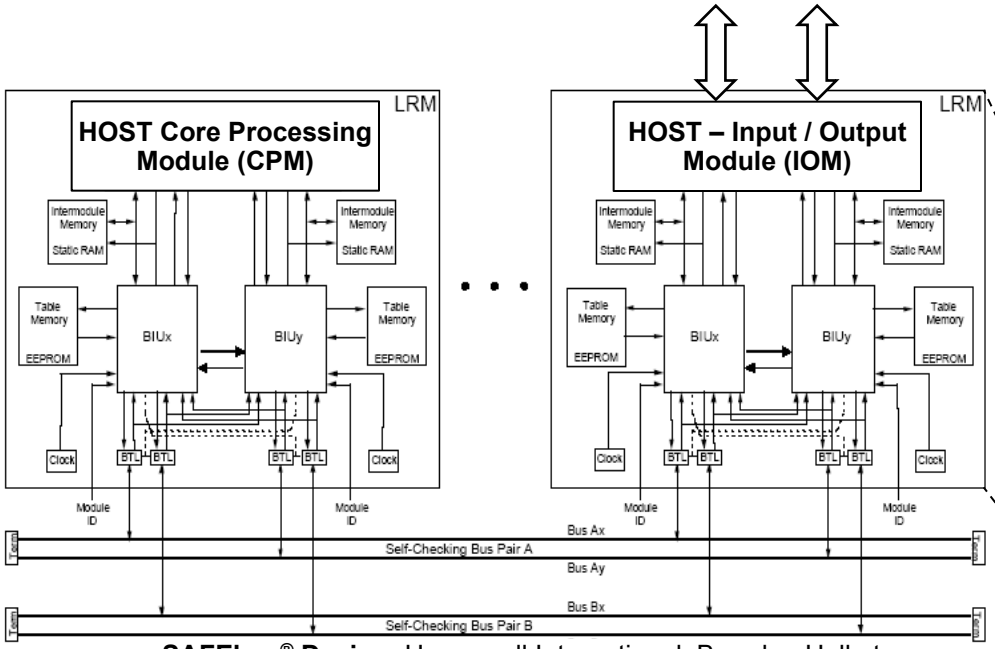


AFDX Network:

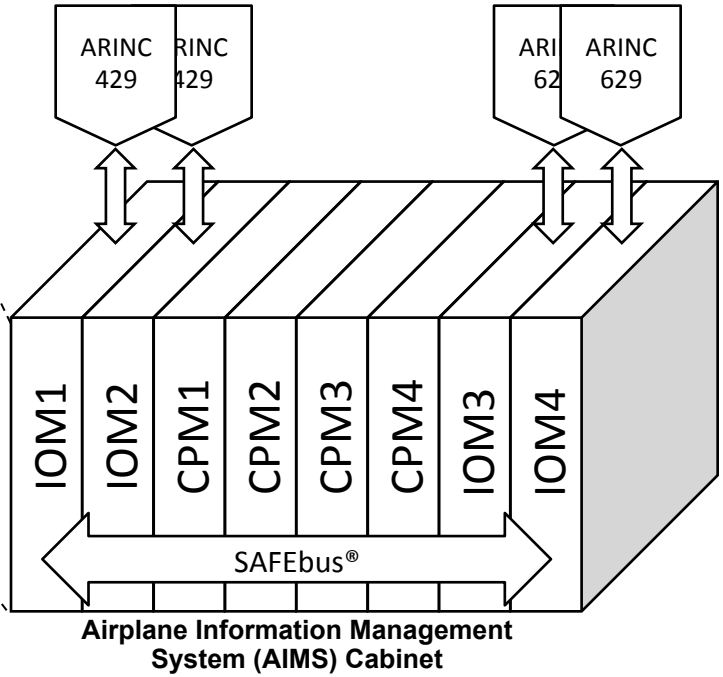
- 100 Mbits
- Redundant Network (A&B) with independent alimentation
- AFDX switches = 2 x 8
- NB of ports (connections) possible on each switch (20-24)
- MTBF of the switch is very high (100 000 hours expected)
- Up 80 AFDX subscriber

Source: http://www.artist-embedded.org/docs/Events/2007/IMA/Slides/ARTIST2_IMA_Itier.pdf

Avionics Hardware Example



SAFEbus® Design: Honeywell International, Brendan Hall et al, "Ringing Out Fault Tolerance." (DSN'05) pp 298-307.



Airplane Information Management System (AIMS) Cabinet

CPM: Core Processing Module
IOM: Input / Output Module
LRM: Line Replaceable Module

BIU: Bus Interface Unit
ARINC 629: communications
ARINC 429: communications

Technical approach

- Modeling the entire aircraft avionics in SAE's AADL open-source standard
- *Formal model* of relevant constraints
- Using a *Satisfiability Modulo Theories* (SMT) solver, extracting information directly from the AADL model.

Innovations:

- Complete, consistent set of constraints defining a *correct* schedule (not an algorithm or a priority scheme) supporting a wide range of architectures and protocols
- Using a generic solving engine (yices SMT solver) to generate schedules
- Solving multiple levels of a hierarchical scheduling problem, all at the same time and in the same model

Phase I Output

As a result of this project, we have generated:

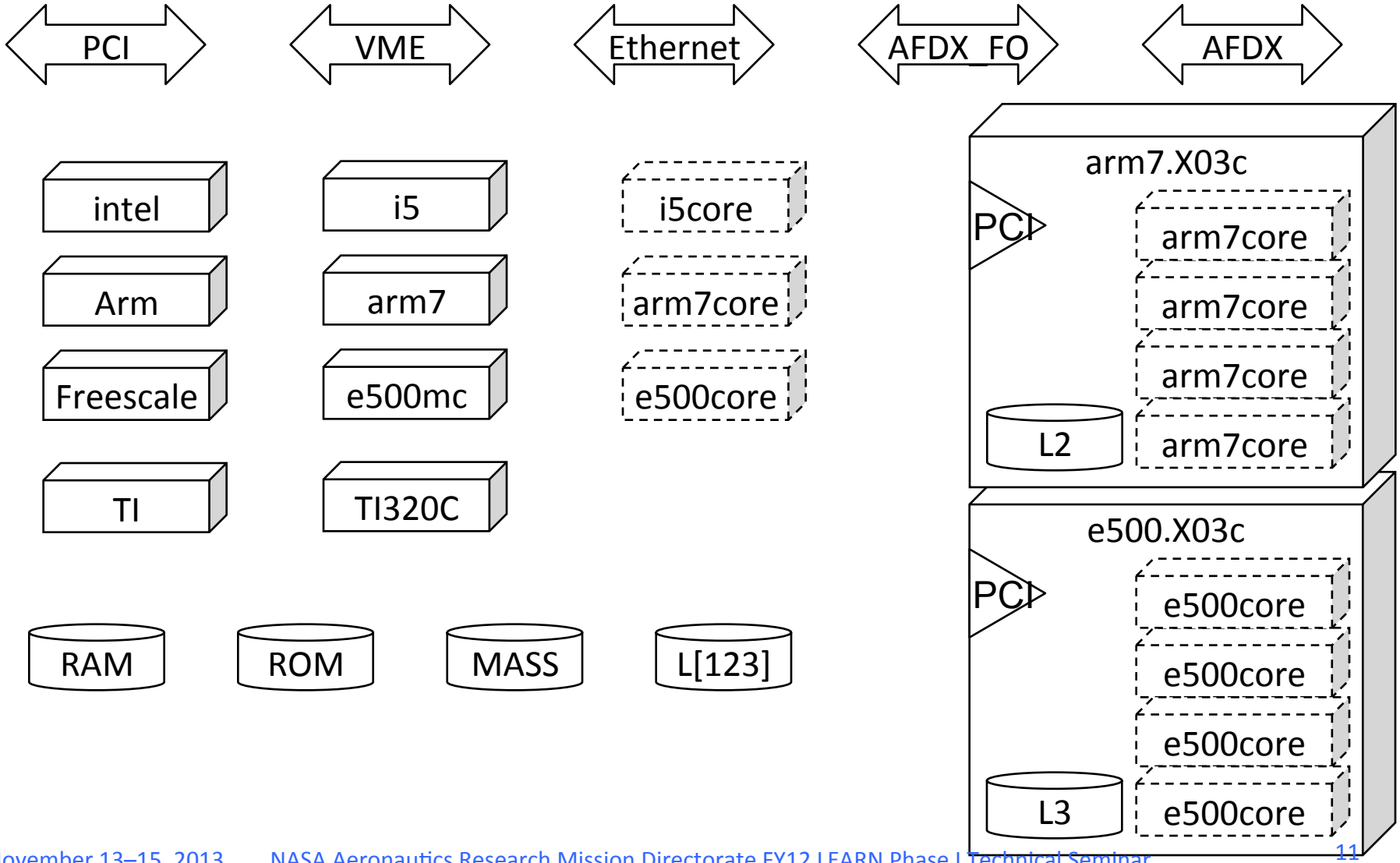
- a formal specification of the complete set of constraints, sufficient to represent a wide range of different avionics architectures and problems,
- a large set of test problems for input to the yices SMT solver, demonstrating the use of those constraints, along with output results and performance data,
- a tunable test problem generator, automatically generating problem instances in yices input format, and
- an exemplar aircraft avionics architecture, rendered in both diagrams and AADL.

These artifacts are available under SBIR data rights for government use. In addition, we intend to turn the forthcoming Phase I final report into one or more technical papers for conference submission.

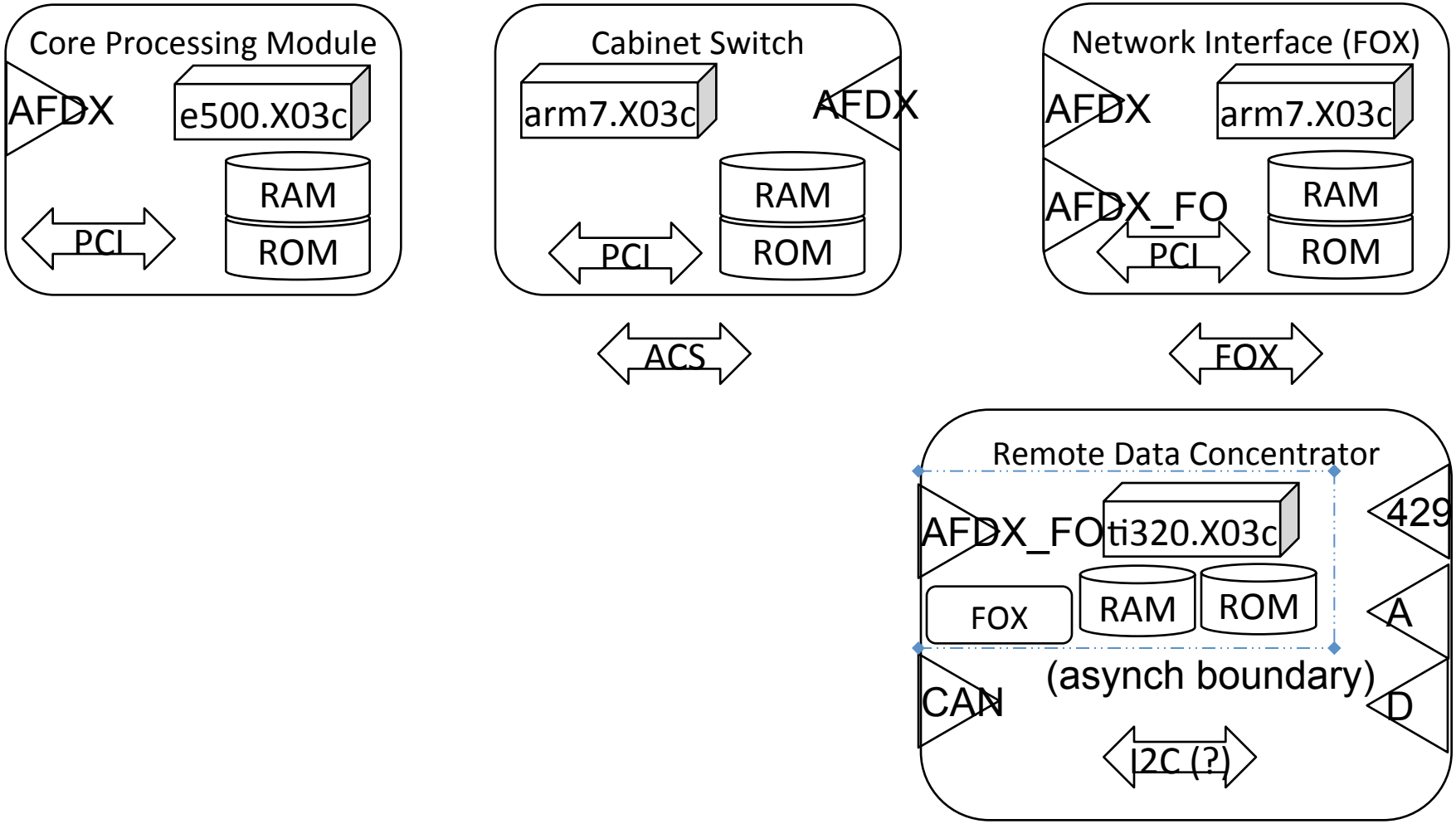
Modeling the entire aircraft avionics

- Hierarchical organization
- Asynchronous boundaries
- ARINC 429, 653, 659, 664
- Varying latencies, rates, criticality
- Shared memory, buffers, and buses
- …

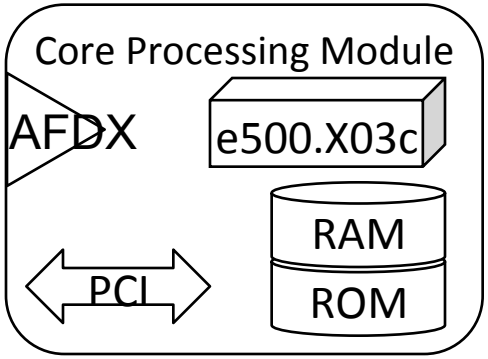
Hardware Parts

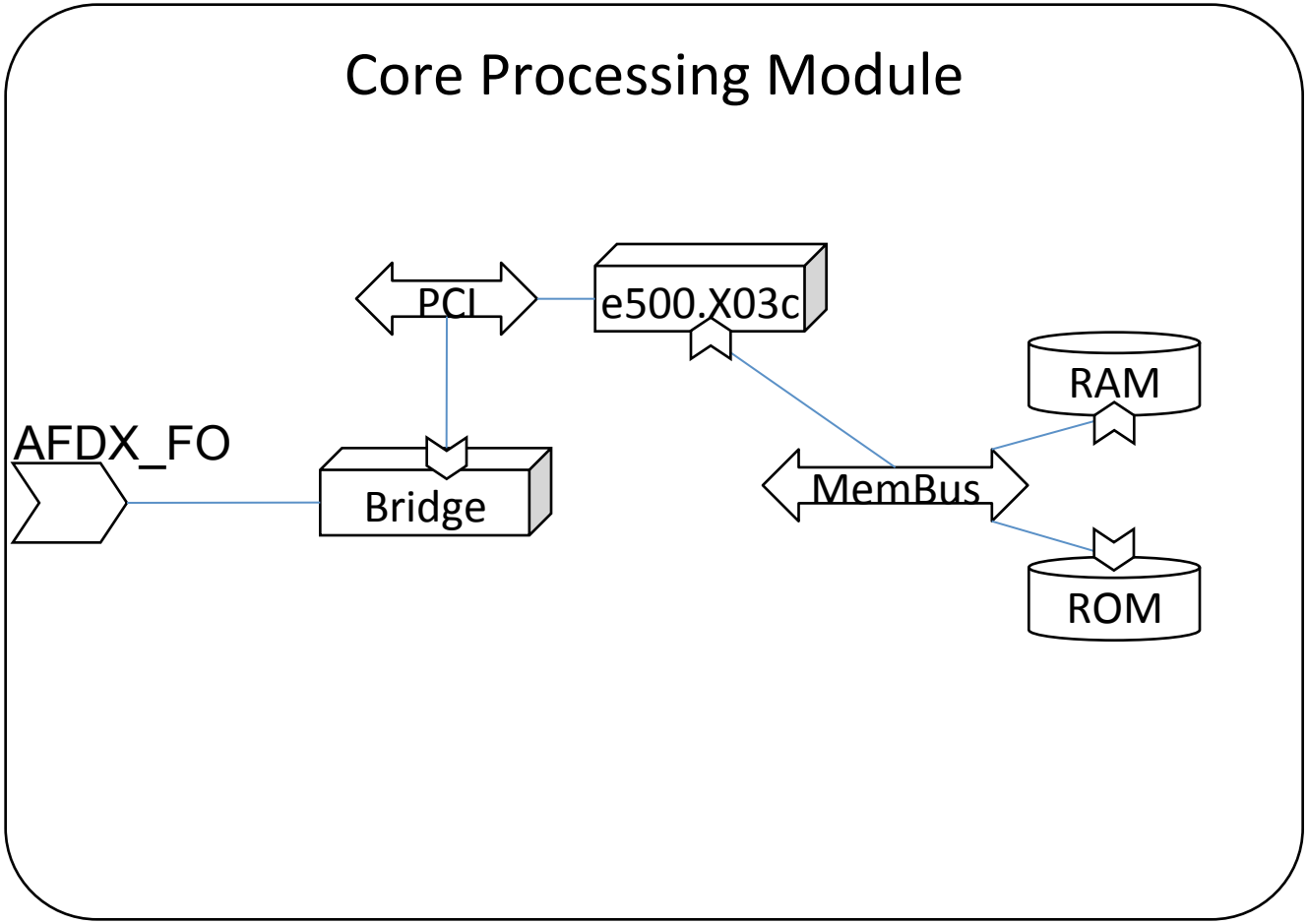


Modules

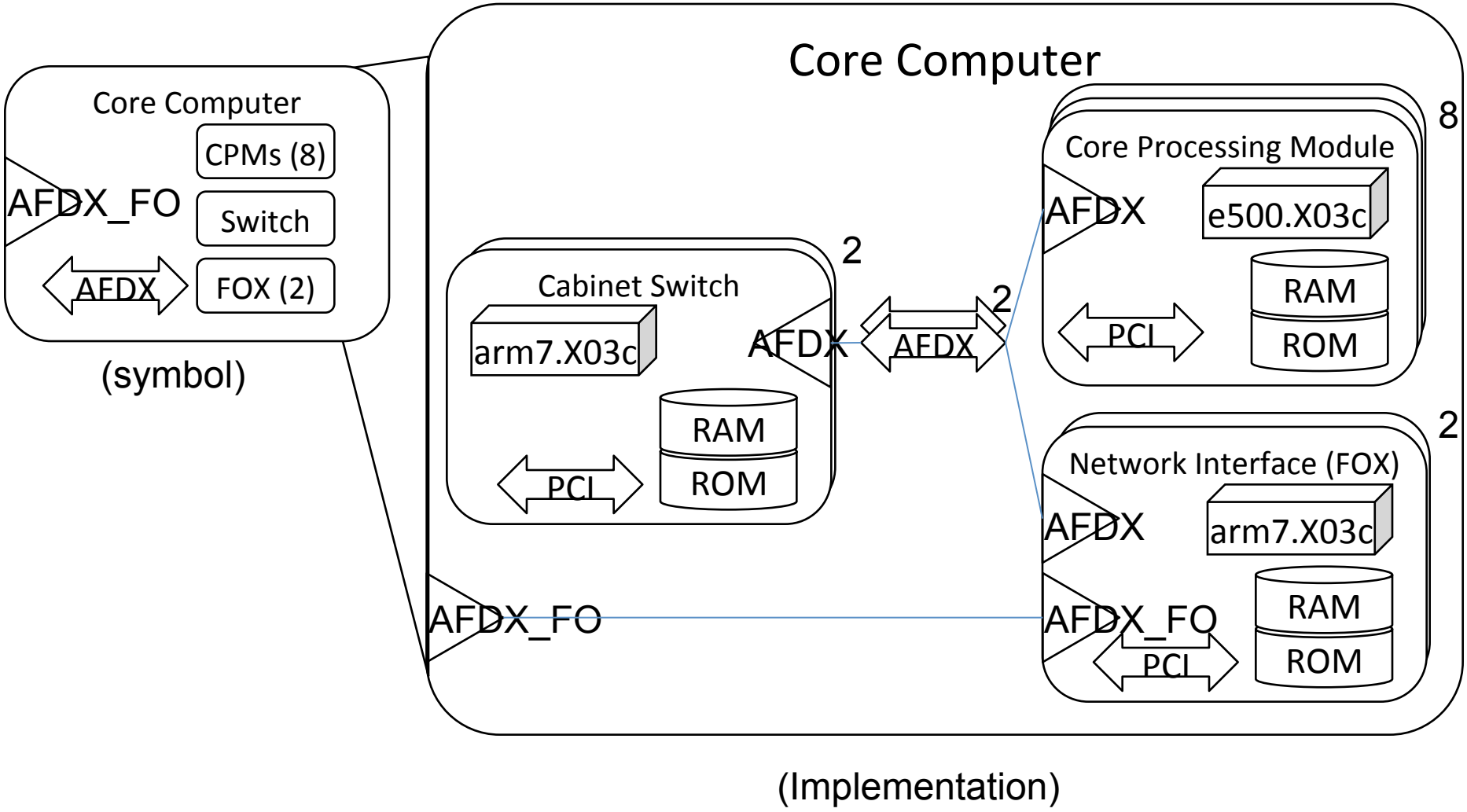

Core Processing Module
AFDX
e500.X03c
RAM
ROM
PCI
Cabinet Switch
arm7.X03c
AFDX
RAM
ROM
PCI
Network Interface (FOX)
AFDX
arm7.X03c
AFDX_FO
RAM
ROM
PCI
ACS
FOX
Remote Data Concentrator
AFDX_FO
ti320.X03c
429
FOX
RAM
ROM
A
(asynch boundary)
CAN
D
I2C (?)

Core Processing Module Detail



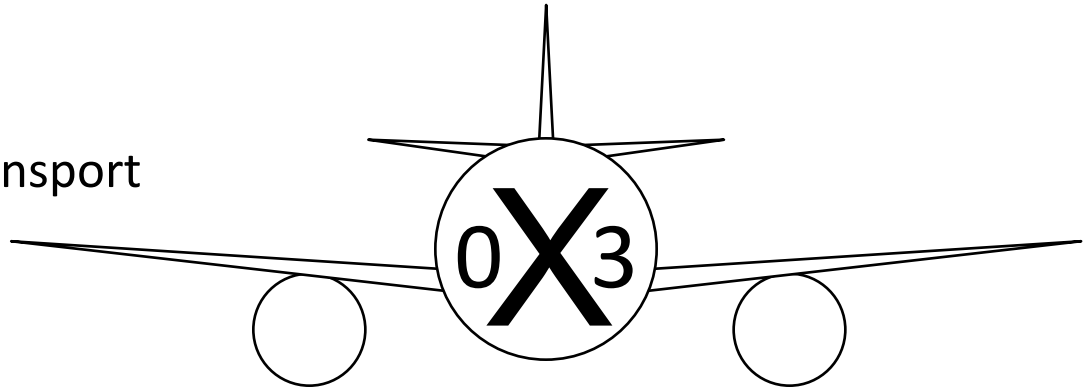


Core Computer

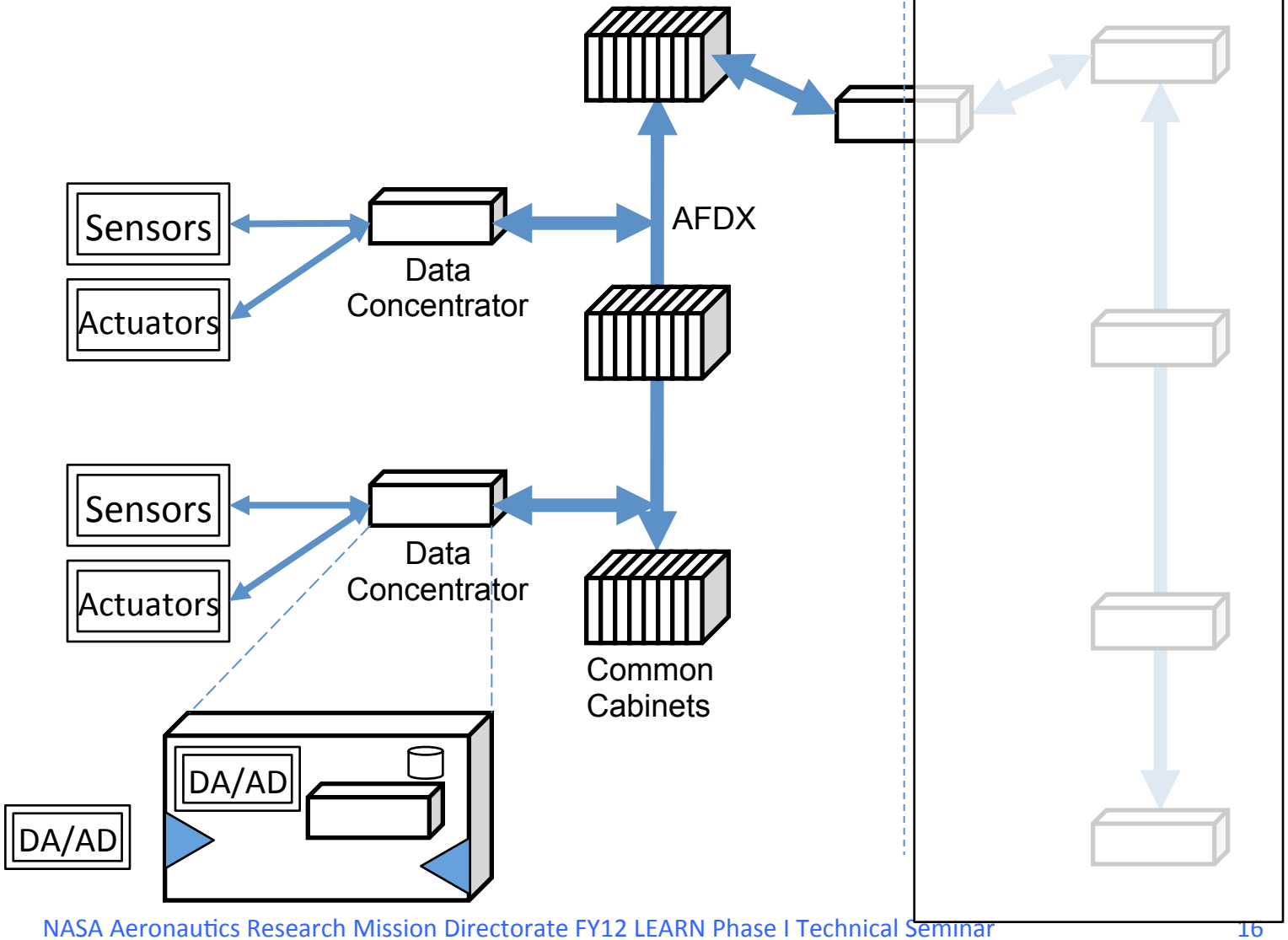


Adventium "X-03c"

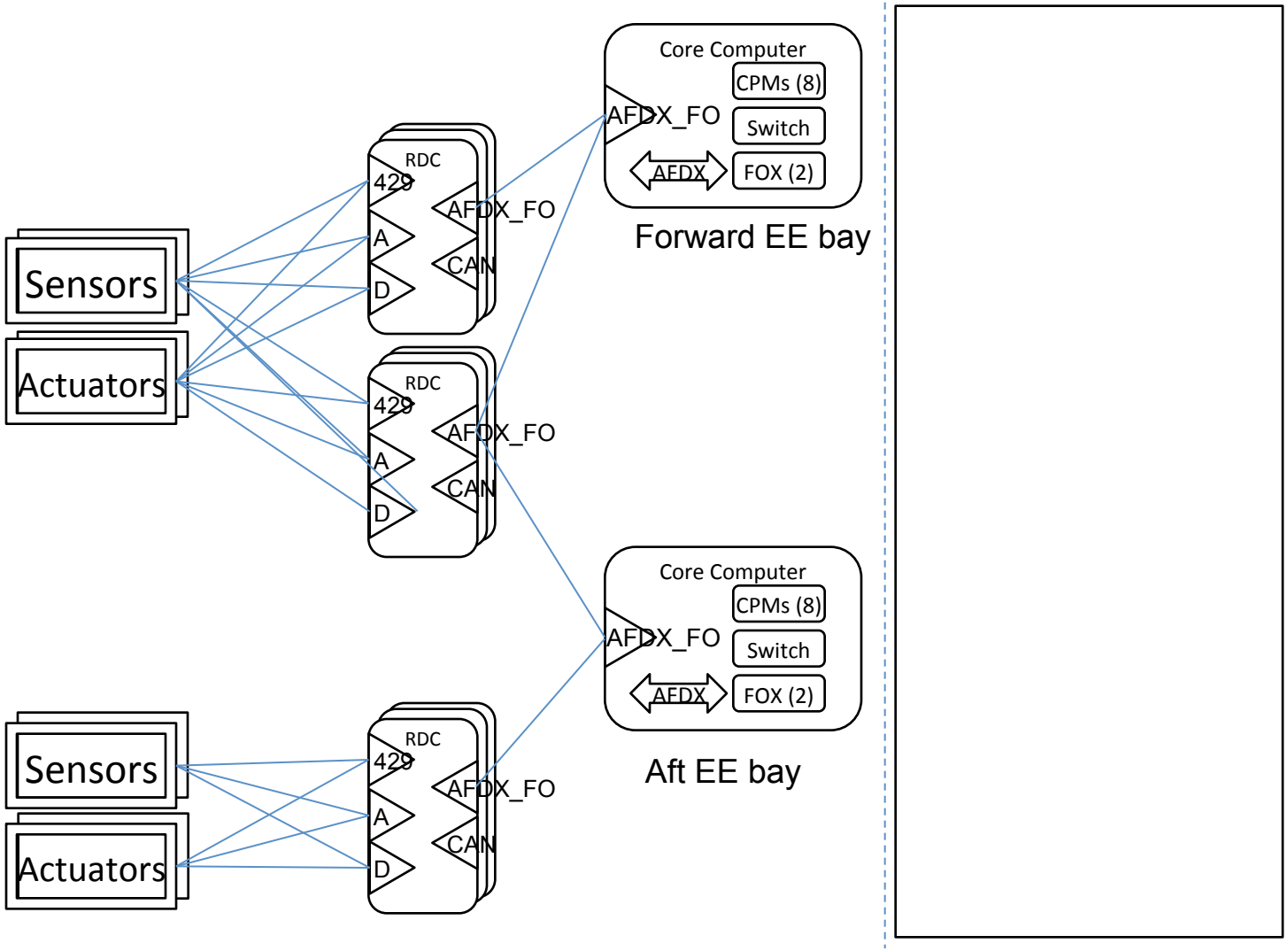
- Next Gen Commercial transport
- Twin engine
- Long haul
- Narrow body
- Mixed passenger / cargo



X-03c Hardware Architecture



X-03c Hardware Architecture (AADL)



Formal model of relevant constraints

Latency

Jitter

Preemption

Over/Undersampling

Asynchronous boundaries

Task grouping/varying context-switch times

Inter- and intra-frame timing constraints

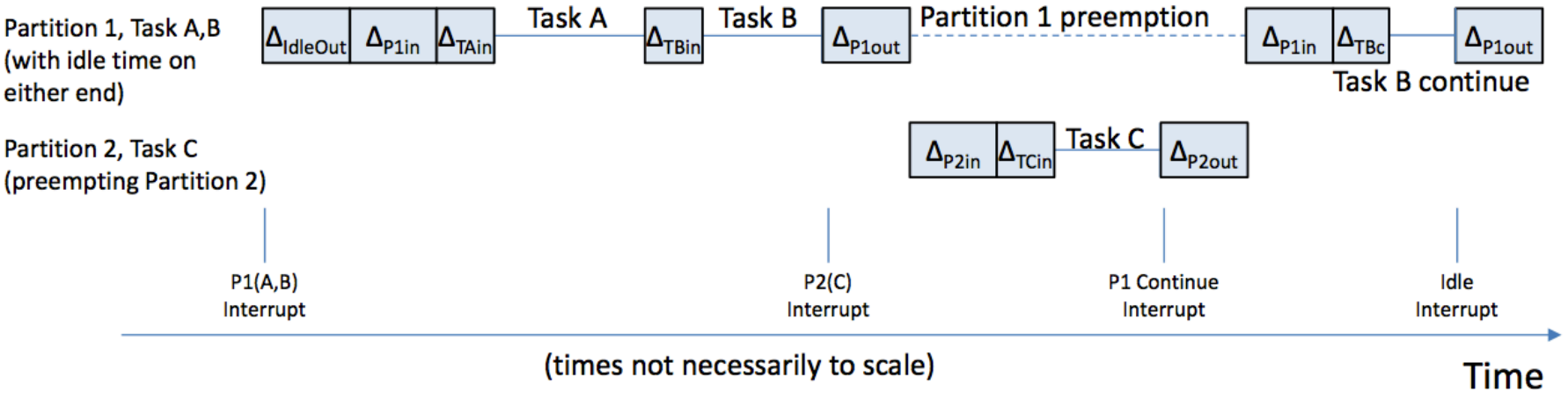
Shared memory

Resource assignment

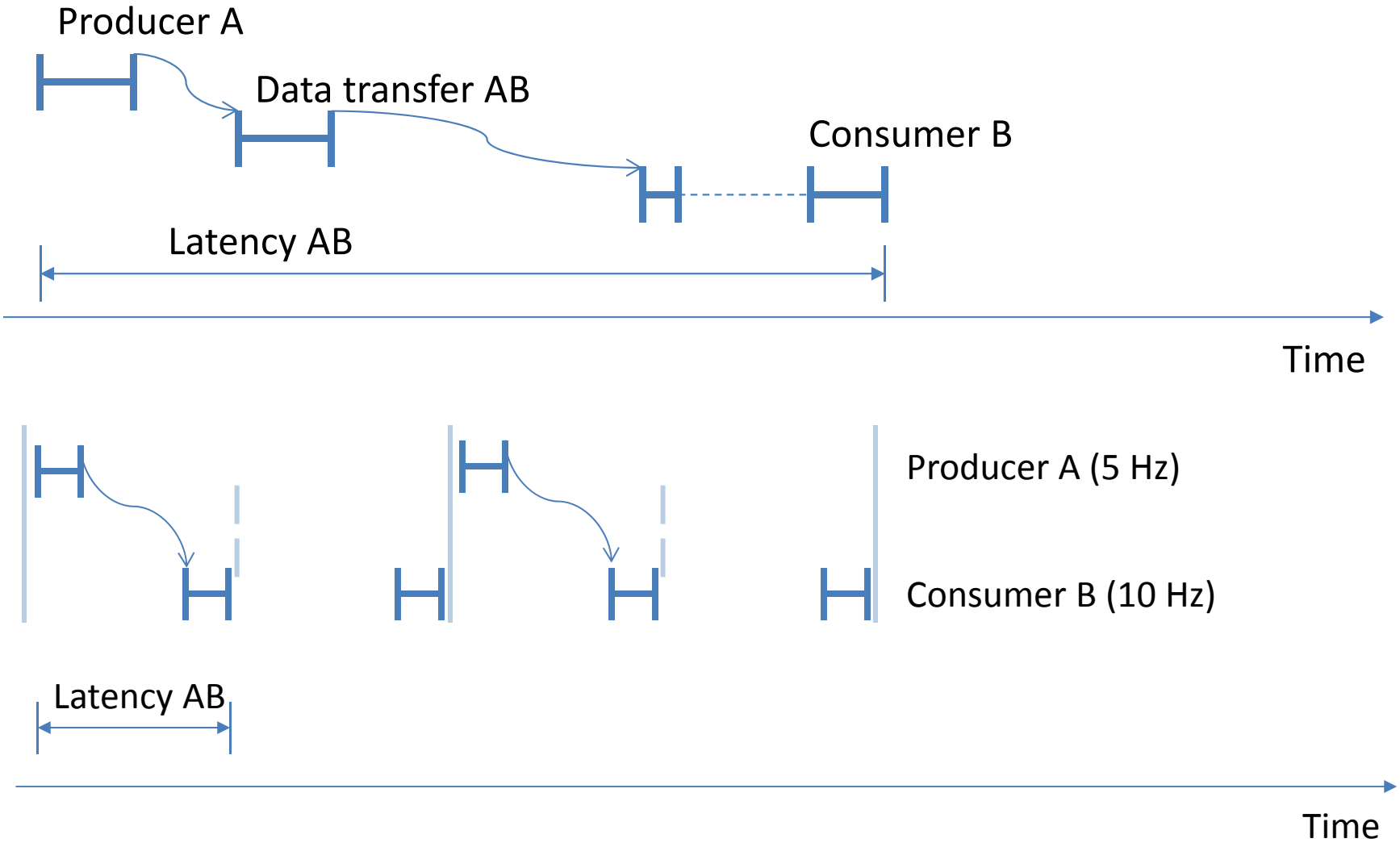
…

For the full set and formal definitions, see the Final Report.

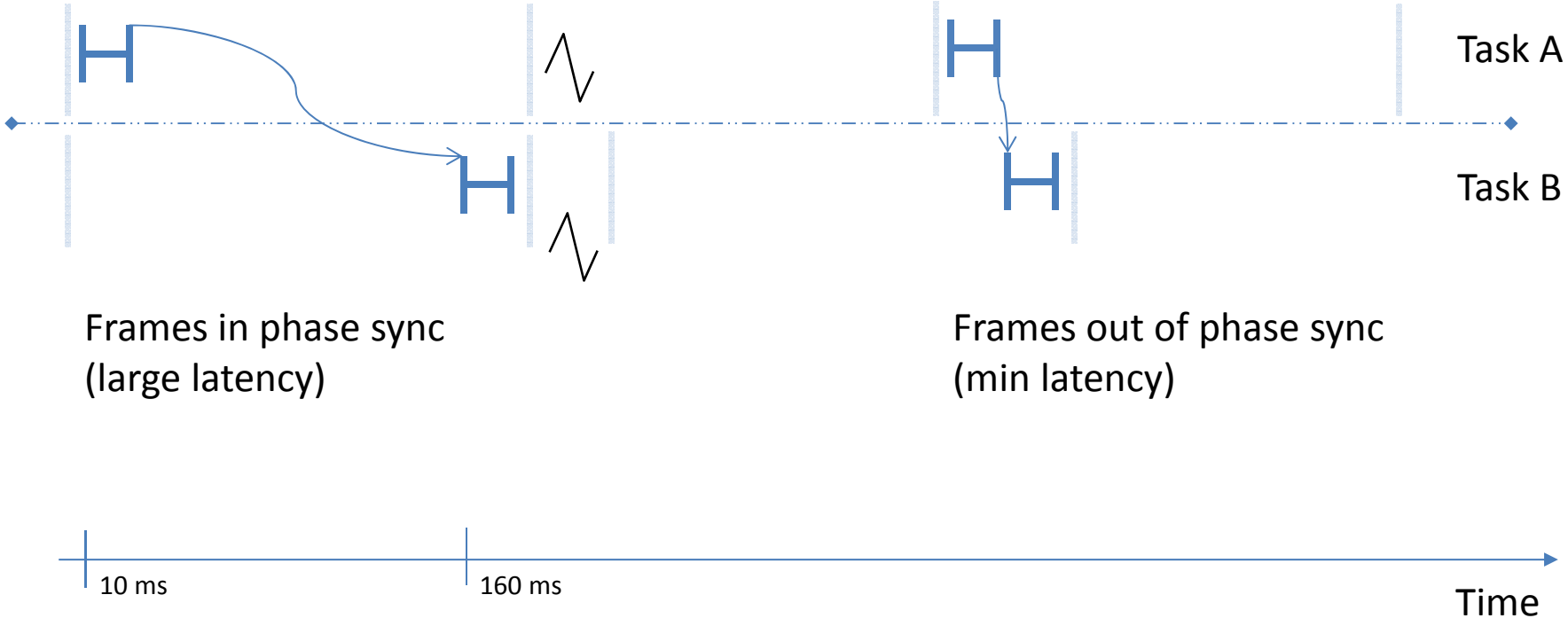
Task Grouping and Preemption



Communication Latency



Latencies Across Asynch. Boundary

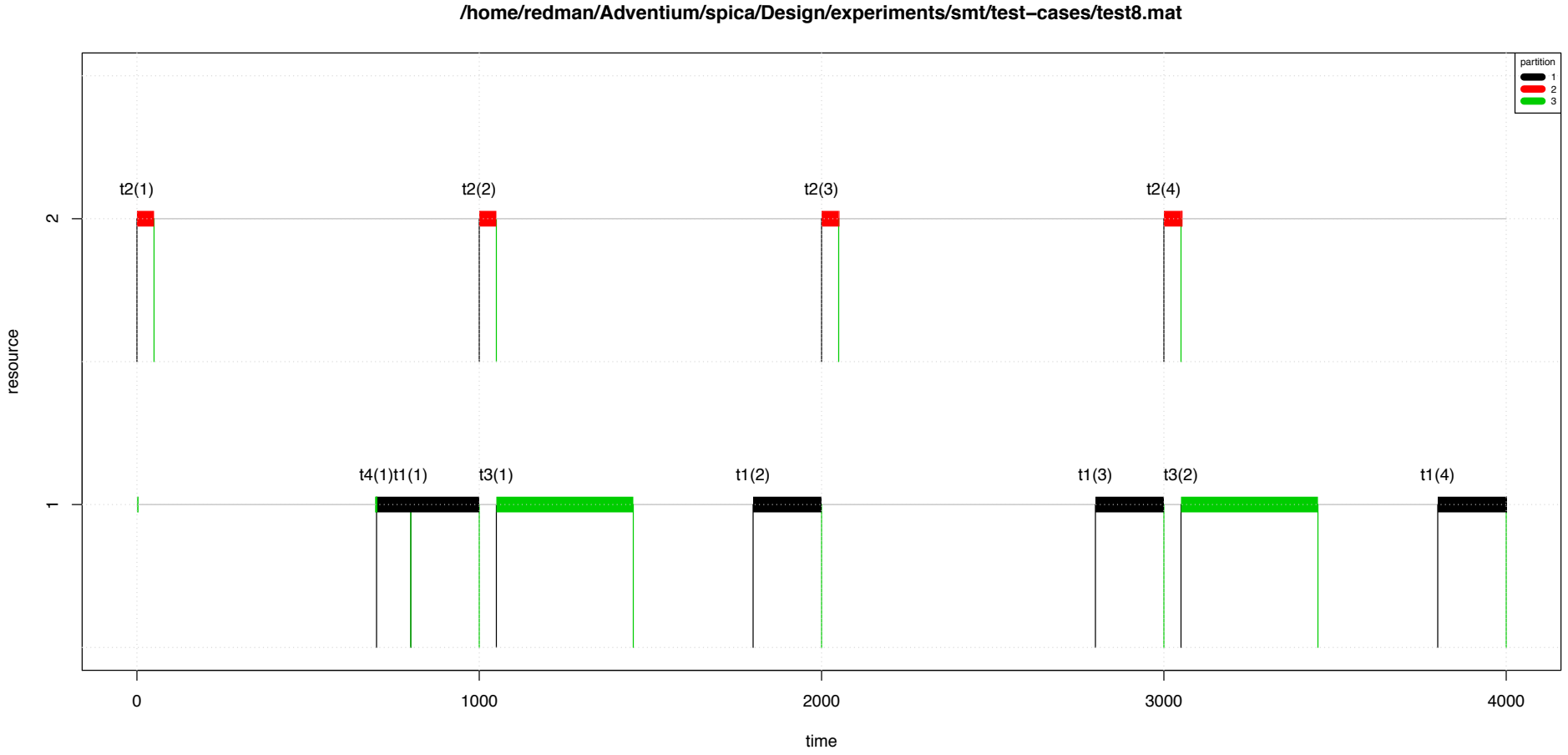


Using a SMT solver

- Effective combination of logical and mathematical reasoning
- Based on technologies demonstrated to scale to millions of variables and constraints

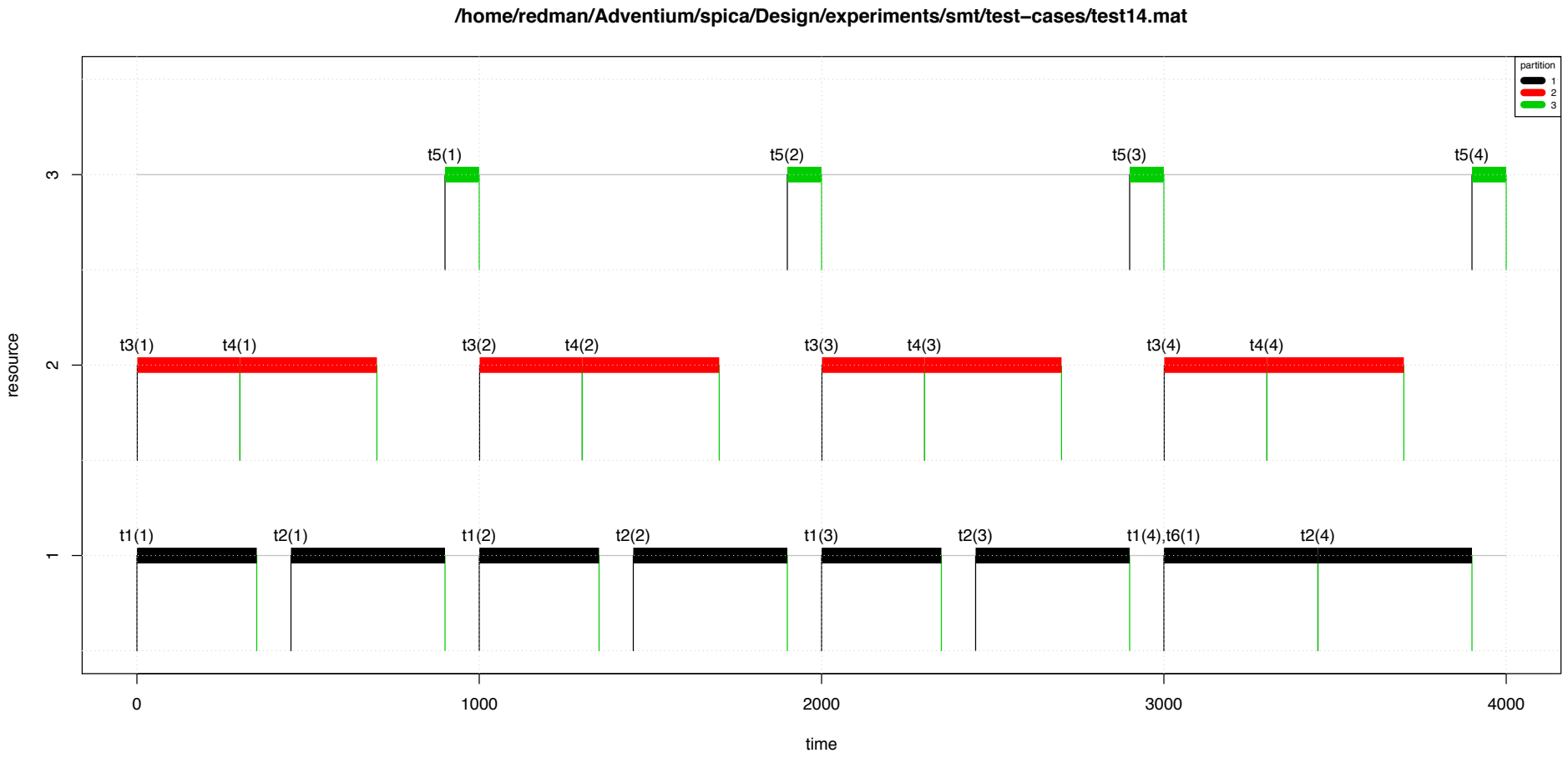
```
;;; Jobs on a given cpu may not overlap
(assert (or (/= pJ1 pJ2) (<= fJ1 sJ2) (<= fJ2 sJ1)))
(assert (or (/= pJ1 pJ3) (<= fJ1 sJ3) (<= fJ3 sJ1)))
(assert (or (/= pJ1 pJ4) (<= fJ1 sJ4) (<= fJ4 sJ1)))
(assert (or (/= pJ1 pJ5) (<= fJ1 sJ5) (<= fJ5 sJ1)))
```

Undersampling

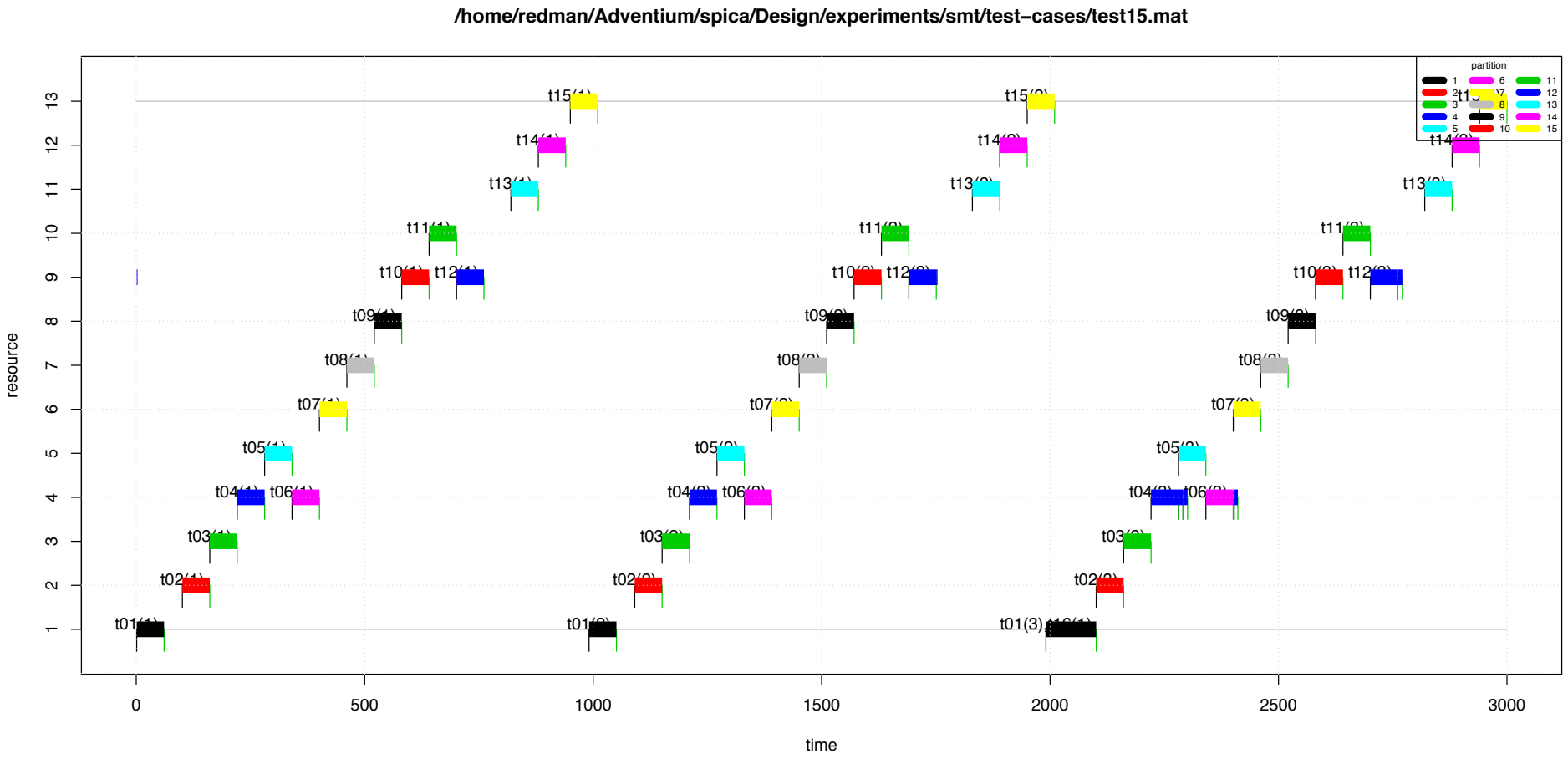


Dataflow and Resource Assignment

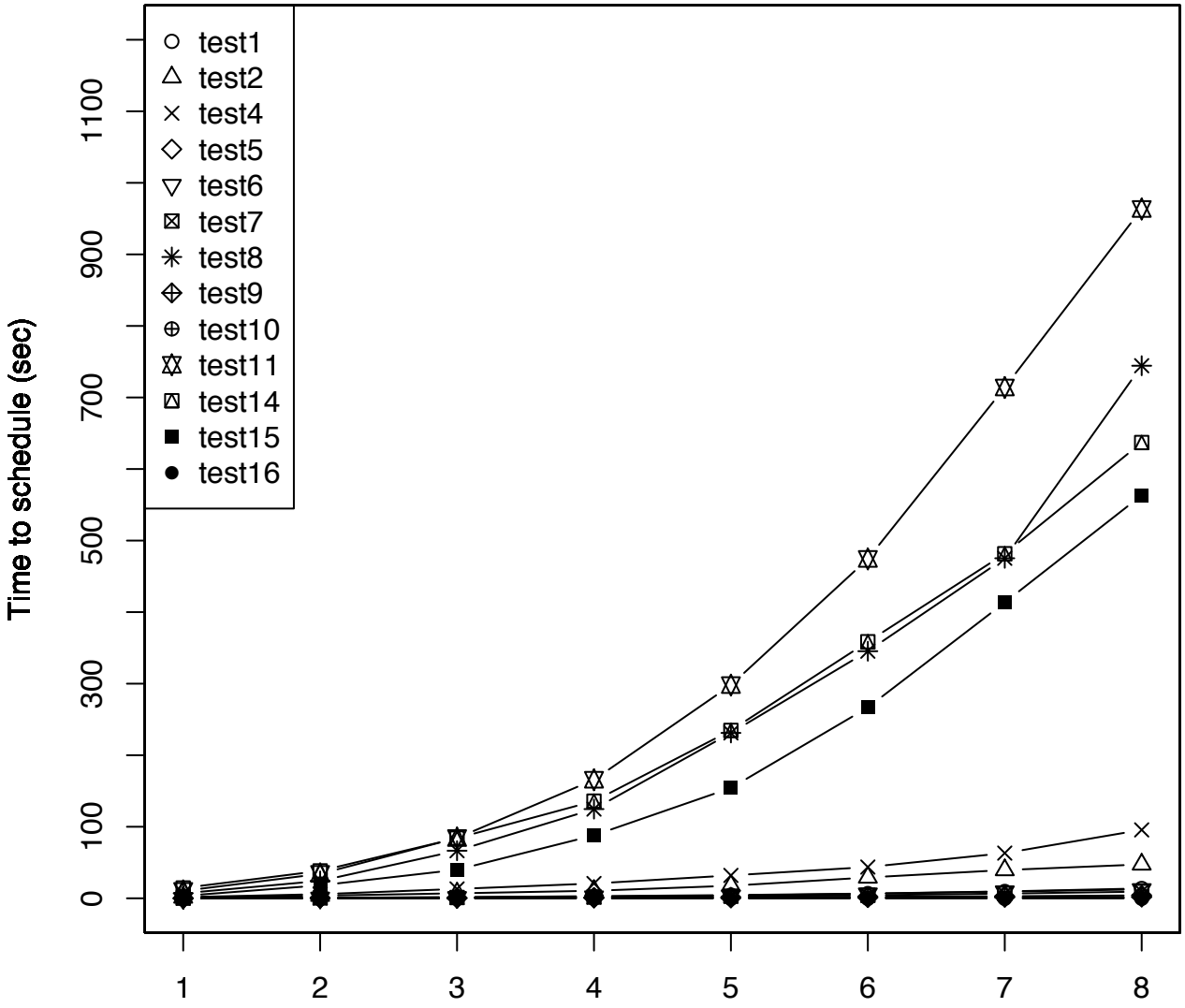
NASA Aeronautics Research Institute



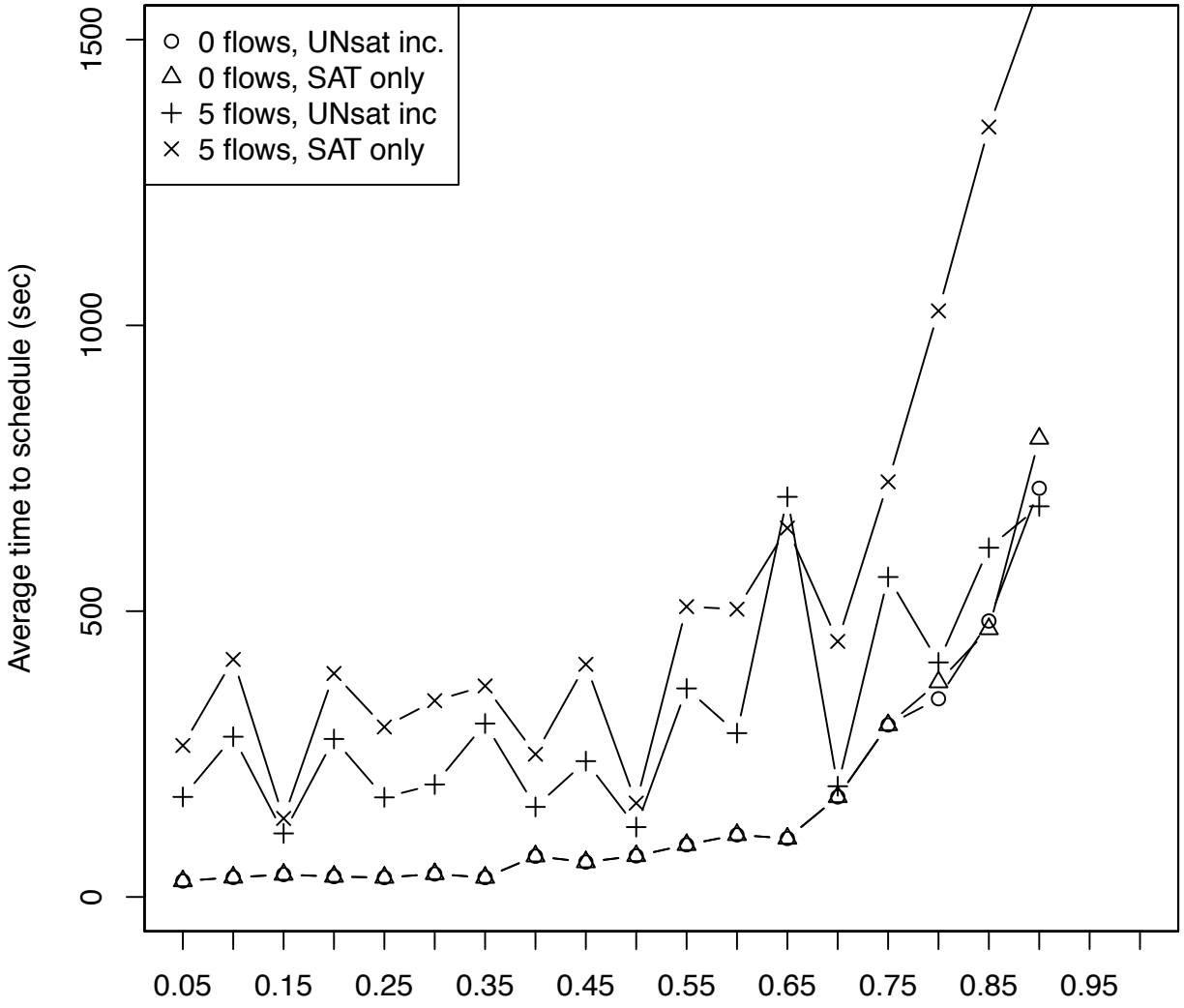
Multiple Buses and Asynch. Boundary



Scaling For Different Problems



Scaling: Processor Load vs. Time



Impact: NASA Programs

- **Integrated Systems Research** -- investigating the avionics-level integration of novel functions, hardware systems, and architectures.
- **Aviation Safety** -- This program includes assurance for flight-critical systems, including managing the complexity of architecting, validating, and verifying the correct functioning of increasingly complex avionics. SPICA's output is a concrete schedule which can easily be verified to satisfy requirements governing execution times, latencies, and sampling rates, as well as more complex issues such as metastable communications across an asynchronous boundary.
- **Orion** -- SPICA is developing relevant capabilities for other complex, networked vehicular systems. For example NASA's Orion MPCV uses several of the protocols and standards SPICA is designed to address.

Impact: Outside of NASA

1. Adventium is part of the System Architecture Virtual Integration (SAVI) consortium as a tool vendor partner. SAVI is an Aerospace Vehicle Systems Institute (AVSI) program, with membership from industry, government, and academia.
2. The Phase I proposal included letters of support from Lockheed Martin and the Army Aviation and Missile Research Development and Engineering Center (AMRDEC).
3. Adventium has a current contract supporting the Army in the development of an Architecture-Centric Virtual Integration Process for Future Vertical Lift mission systems.

Next steps

Technical issues

- Multi-core, more generally, e.g., contention for on-board cache
- Integrating multiple scheduling approaches
- Other protocols, as needed

Maturation

- Scaling
- Finalize translation from AADL to SMT input format
- Different avionics architectures

Multi-core

- Characteristics addressed in Phase I
  - Shared IO
  - Shared buffers
  - Task allocation to discrete processing resources
  - (A)synchronous communicating processes
- Issues deferred to Phase II
  - Memory contention from different cores, including various levels of cache
  - Migration between cores
  - (virtualization)

Integrating Different Schedulers

- SPICA produces a *static schedule* that is mathematically guaranteed to satisfy the input constraints.
- Dynamic schedulers are specified in terms of a set of *schedulability constraints.*
- Current integrations provide a static allocation within which the dynamic scheduler(s) have control.
- SMT is specifically designed to incorporate specialized types of constraints

So: Is it possible to specify schedulability constraints in a decomposable form, such that the resulting allocation may take several forms, but is in any case guaranteed to be schedulable? For example, is one allocation more efficient than another at accommodating sporadic, high-priority, low-latency tasks and still providing the required guarantees for other tasks?

Scaling

- Current system solves problems involving dozens to hundreds of constraints, in minutes.
- SMT technology has demonstrated performance on *millions* of constraints
- Growth in solving time with problem size is reasonable.

Next steps:

- Search control
- Problem reformulation

In previous work, we have demonstrated several orders of magnitude increase in problem size and decrease in solving time.

The End